REPORT DOCUMENTATION PAGE

FORM APPROVED
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED

4. TITLE AND SUBTITLE OF REPORT
Modular implementation of efficient self-checking checkers for the Berger code.

5. FUNDING NUMBERS
Grant No. N00014-94-1-0962

6. AUTHOR(S) P.K. Lala & A. Walker

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
North Carolina A&T State University
Dept. of Electrical Engineering
Greensboro, NC 27411.

8. PERFORMING ORGANIZATION REPORT NUMBER:

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Office of Naval Research

10. SPONSORING/MONITORING AGENCY REPORT NUMBER:

11. SUPPLEMENTARY NOTES:

12a. DISTRIBUTION AVAILABILITY STATEMENT
UNLIMITED

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

The Berger code is the least redundant systematic code available for detecting single and unidirectional multibit errors. In this report, a technique for designing efficient checkers for the Berger code is presented.

14. SUBJECT TERMS	15. NUMBER OF PAGES: 20
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT:	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT

Progress Report

Grant No: N00014-94-1-0462

Office of Naval Research

~~Self-Checking State Machine Realization in CMOS~~

Reporting Period : January 1 --- March 31, 1996

Dr. P.K. Lala
Principal Investigator
Dr. A. Walker
Co-Principal Investigator
Department of Electrical Engineering
North Carolina A&T State University
Greensboro, NC 27411.

Modular Implementation of Efficient Self-Checking Checkers for the Berger Code

The Berger code is the least redundant systematic code available for detecting single and multibit unidirectional errors [1]. The number of check bits, k, for a conventional Berger code is determined by the equation : $k = \lceil \log_2 (I+1) \rceil$,where I is the number of information bits. There are two encoding schemes, B0 and B1, used to encode information in conventional Berger code. In the B0 encoding scheme, the check bits are the binary representation of the number of 0's in the information vector, whereas in the B1 encoding scheme, the check bits are the bit by bit complement of the binary representation of the number of 1's in the information vector. If the number of information bits is related to the number of check bits by the relationship, $I = 2^k - 1$, then the resulting code is called the maximal length Berger code.

In this report, a technique for designing efficient checkers for Berger codes is presented. Figure 1 shows the block diagram of the checker. Information bits are passed through the check bit generator. The resulting check bits are then compared to the predicted check bits by a two-rail checker. If there is no fault present in the two-rail checker or the check bit generator, the outputs of the checker are complementary (i.e. 01 or 10), otherwise they are 00 or 11. We will focus only on the implementation of the check bit generator (CBG) circuit. The CBG is designed assuming the B1 encoding scheme for non-maximal length Berger codes, and the B0 encoding scheme for maximal length Berger codes. The basic building block of the proposed CBG is a 4 input *1's counter* [see Figure 2]. This 4-input 1's counter receives four inputs and generates at its three output bits the binary representation of the number of 1's in the four input bits.

Two other types of *1's counters*, having 2 and 3 inputs, are used to realize checkers for variable length information bits. Figure 3 shows the schematics of these counters. A 1's counter of any size can be constructed from the basic counters. For example, a 5 input 1's counter can be constructed from a 3 input 1's counter, a 2 input 1's counter and a standard 2-bit adder.

A CBG is implemented with two 1's counters and an addition array. The information bits are partitioned into two blocks I_r and I_{r^*} . The I_r block contains $\lceil I/2 \rceil$ bits and the I_{r^*} block contains $I-\lceil I/2 \rceil$ bits. The size of the 1's counters needed to implement a CBG depends on the number of bits in blocks, I_r and I_{r^*} . If the number of information bits in a block are 4 or fewer, an appropriate 1's counter (shown in Figs. 2, 3(a) and 3(b)) in conjunction with the appropriate addition array are used to complete a design. However, if the number of bits in a block is greater than 4, the block is partitioned into sub-blocks. The partitioning process is continued until a sub-block can be implemented with a pre-designed 1's counter and an addition array. The counter corresponding to a sub-block is used as a building block to implement larger size 1's counters. The number of 1's in blocks I_r and I_{r^*} derived by partitioning I information bits is represented by vectors r and r*, respectively. The number of bits in r and r* is $\log_2 (\lceil I/2 \rceil + 1)$ and $\log_2 ((I - \lceil I/2 \rceil) + 1)$, respectively. The check bits, k , for I information bits is the sum of r and r*, and is derived by adding two bits at a time as shown below:

$$r = r_n \; r_{n-1} \; \ldots\ldots \; r_3 \; r_2 \quad r_1 \; r_0$$

$$r^* = r^*_n \; r^*_{n-1} \; \ldots\ldots \; r^*_3 \; r^*_2 \quad r^*_1 \; r^*_0$$

$$k = k_n \; k_{n-1} \; \ldots\ldots \; k_3 \; k_2 \quad k_1 \; k_0$$

If there are 2^n (where n is an integer) inputs to a block, the block produces a vector r (or r*) whose most significant bit is considered to be a *special* most significant bit (SMSB). If the SMSB of r or r* is 1, it is the only '1' present in the vector, and a carry is not produced when r and r* are added.

To illustrate, let us assume:

I=7;
then $\mathbf{I_r = 4}$; and $I_{r^*} = 3$;
length of r: 3 bits $\Rightarrow \mathbf{r_2}\, r_1\, r_0$
length of r*: 2 bits $\Rightarrow r^*_1\, r^*_0$

If a vector consisting of all 1's is present at the inputs to the 4 input counter (I_r = 4), the most significant bit of r, i.e. r_2, will be 1 and r_1 and r_0 will be 0. Irrespective of the values of r^*_1 and r^*_0, summing them with r_1 and r_0 will never produce a carry.

The two least significant bits of vectors r and r* (i.e. r_0,r^*_0,r_1, and r^*_1) are grouped together in a set called a *primary* group. A 2-bit adder is used to sum these bits. The outputs of the adder become the least significant check bits, i.e. k_1 and k_0. The carry-out generated during this addition generally propagates to a secondary group. In some cases, the carry becomes the check bit k_2. To illustrate, let us assume:

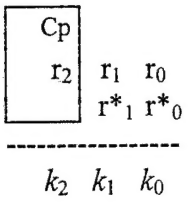
$I \Rightarrow 10110$. $I_r \Rightarrow 101$; $I_{r^*} \Rightarrow 10$; therefore, $r \Rightarrow r_1\, r_0 = 10$; $r^* \Rightarrow r^*_1\, r^*_0 = 01$.

The check bits, *k*, are obtained by adding r and r* as shown in Figure 4. The special 2-bit adder is a derivative of the standard 2-bit adder used in all other CBG designs (Figure 5(a)). Since $r_1 r^*_1 = 11$ does not occur in a CBG design for (I,k) = (5,3), gates with bits $r_1 r^*_1$ in Figure 5(a) are discarded to form the special 2-bit adder (Figure 5(b)). The special 2-bit adder is used in a sub-block which has five information bits. Otherwise, as a default, the standard 2-bit adder is used to sum the bits in the primary grouping.

A *secondary* group, unlike the primary group, may contain as many as 4 bits or as few as 1 bit. Maximal length information bits (I = {7,15,31,....}) will result in an odd number of bits in the secondary group whereas non-maximal length information bits will result in an even number of bits. The type of adder to be used to sum the bits of the secondary group is dependent not only upon the number of bits in the secondary group but also whether or not the most significant bit of r (or r*) is a SMSB. When the information bits is non-maximal, the most significant bit of r (or r*) may or may not be a SMSB. However, the most significant bit of r is always a SMSB if the information bits are maximal.

We will consider several cases with different numbers of bits in the secondary group. For simplicity, only a single secondary group will be considered.

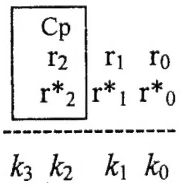
Case 1: One bit in the secondary group.



Note that this case results only when I=7; therefore, r_2 is a SMSB. The carry circuit of the standard 2-bit adder (Figure 5(a)) is modified to accommodate r_2 and is shown in Figure 7. The sum circuit of the modified 2-bit adder is the same as that of the standard 2-bit adder. The CBG block diagram for (I,k) = (7,3) is shown in Figure 6. To illustrate, let us assume:

$I \Rightarrow 1111101$; $I_r \Rightarrow 1111$; $I_{r^*} \Rightarrow 101$; therefore,
$r = 100$; $r^* = 10$; $k = 110$.

Case 2: Two bits in the secondary group.



If bits, r_2 and r^*_2, are SMSB's[Figure 8], a Type I adder is used to generate check bits, k_3 and k_2. Otherwise, a full adder is used. Figures 9 and 10 show the realizations of the Type I adder and the full adder, respectively.

Let us illustrate Case 2 by designing a CBG for (I,k) = (11,4):

$I_r = 6$ ($\lceil I/2 \rceil$) bits and $I_{r^*} = 5$ ($I - \lceil I/2 \rceil$) bits; therefore $r = r_2\, r_1\, r_0$; $r^* = r^*_2\, r^*_1\, r^*_0$; and $k = k_3\, k_2\, k_1\, k_0$.

The implementation of the CBG for (I,k) = (11,4) is shown in Figure 11. Note that neither r_2 nor r^*_2 is a SMSB; therefore a full adder, rather than a Type I adder is used to sum the bits in the secondary group.

Case 3: Three bits in the secondary group.

C_p			
r_3	r_2	r_1	r_0
	r^*_2	r^*_1	r^*_0
k_3	k_2	k_1	k_0

This case results only when I = 15; therefore, r_3 is a SMSB[see Figure 12]. The full adder's carry circuit is modified to generate check bit k_3. The modification of the full adder's carry circuit is shown in Figure 13.

Case 4: Four bits in the secondary group.

	C_p		
r_3	r_2	r_1	r_0
r^*_3	r^*_2	r^*_1	r^*_0
k_4	k_3	k_2	k_1	k_0

If both bits, r_3 and r^*_3 are SMSB's, then a Type II adder is used[Figure 14]. Otherwise, a parallel 2-bit adder is used[Figure 15]. The circuit realization of the Type II adder is shown in Figure 16. The parallel 2-bit adder consists of two 2-bit adders; and three 2-to-1 multiplexers; with one of the adders assuming a carry-in of 1. The select lines of the multiplexers are driven by the carry-out of the primary group. For instance, the outputs of the 2-bit adder (with $C_p = 1$) is chosen when the primary group produces a carry-out of 1. The block diagram of the parallel 2-bit adder and the realization of the 2-bit adder (with $C_p = 1$) used in the parallel 2-bit adder are shown in Figures 17 and 18, respectively.

If a secondary group has 4 bits, it is treated as a primary group. The selection of adder modules needed to add bits in subsequent secondary groups is chosen using the same guidelines described in Cases 1 - Case 4. However, two special cases may arise when more than one secondary group are considered:

Case 5: One bit in the secondary group2.

C_s				
r_4	r_3	r_2	r_1	r_0
	r^*_3	r^*_2	r^*_1	r^*_0
k_4	k_3	k_2	k_1	k_0

In this case, the parallel 2-bit adder configuration is modified to accommodate the SMSB, r_4. In the parallel 2-bit adder[Figure 17], the standard 2-bit adder is replaced by the modified 2-bit adder discussed previously in Case 1. Bit, r_4, becomes an input to the modified 2-bit adder.

Case 6: Two bits (both of which are SMSB's) in the secondary group2.

C_S				
r_4	r_3	r_2	r_1	r_0
r^*_4	r^*_3	r^*_2	r^*_1	r^*_0
k_5 k_4	k_3	k_2	k_1	k_0

Since r_4 and r^*_4 are SMSB's, a Type III adder [Figure 19] is used to generate check bits k_5 and k_4. Note that in this case, carries, C_0 and C_1, generated by the parallel 2-bit adder, propagate to the subsequent secondary group. Otherwise, the procedure given in Case 2 requires the full adder to sum bits C_S, r_4, and r^*_4. Figure 20 shows a block diagram of the CBG design for $(I,k) = (32, 6)$.

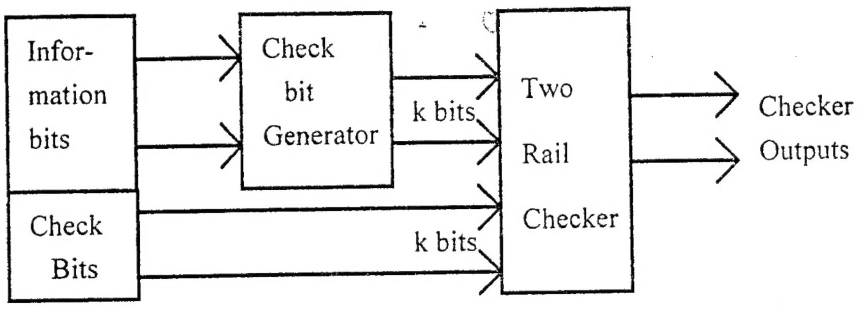

Infor-
mation
bits
Check
Bits
Check
bit
Generator
k bits
k bits
Two
Rail
Checker
Checker
Outputs

Figure 1

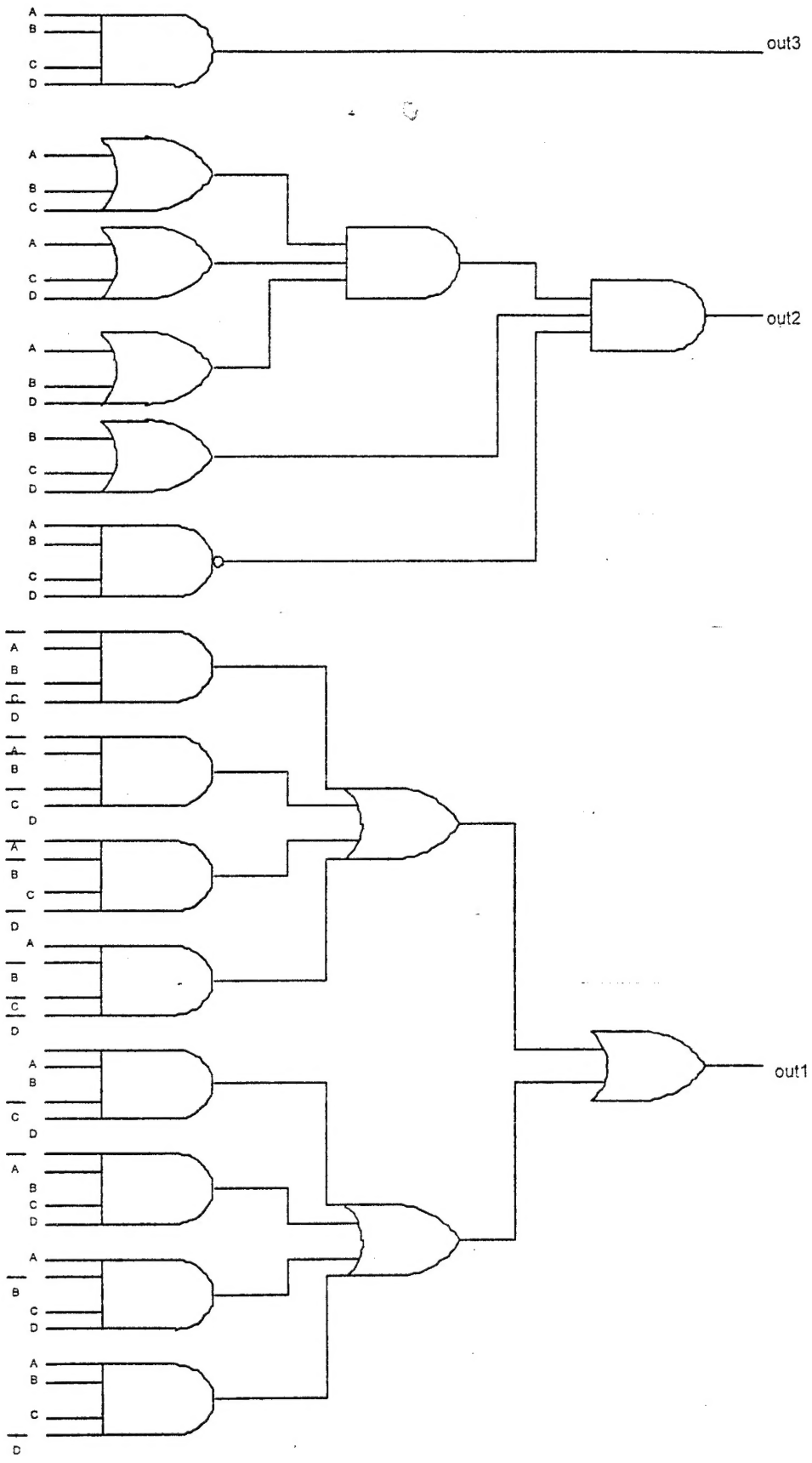


Figure 2 4 input 1's counter

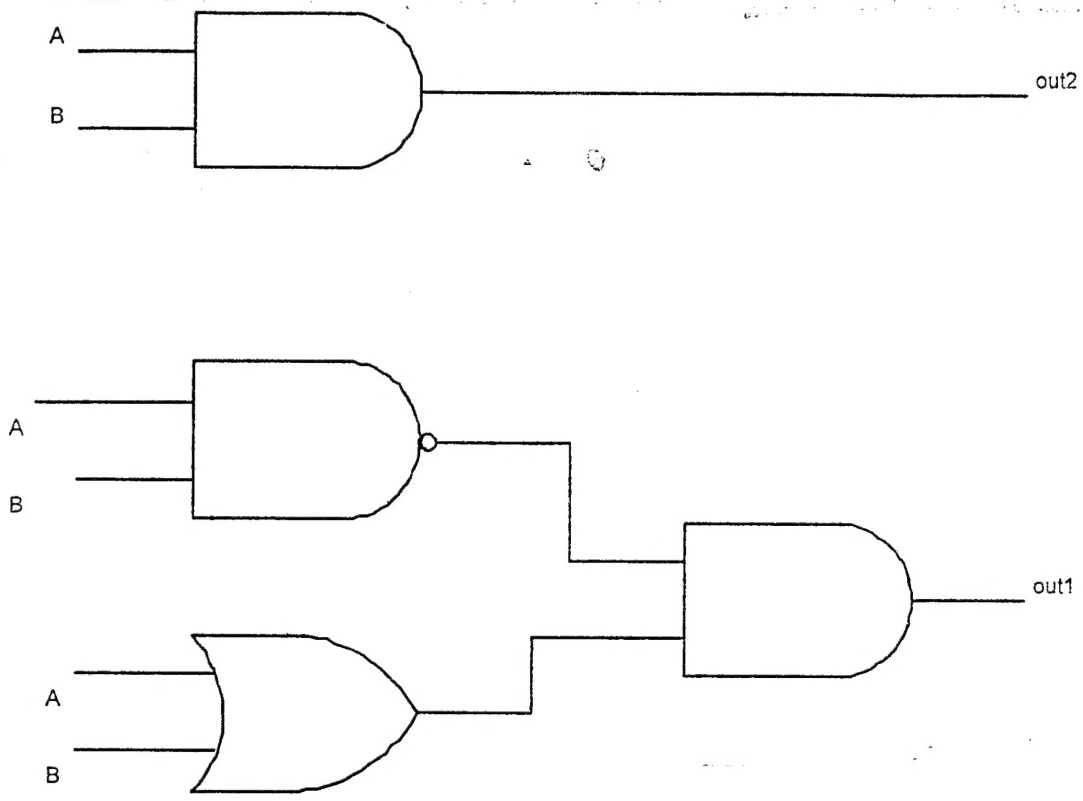


Figure 3(a) 2 input 1's counter

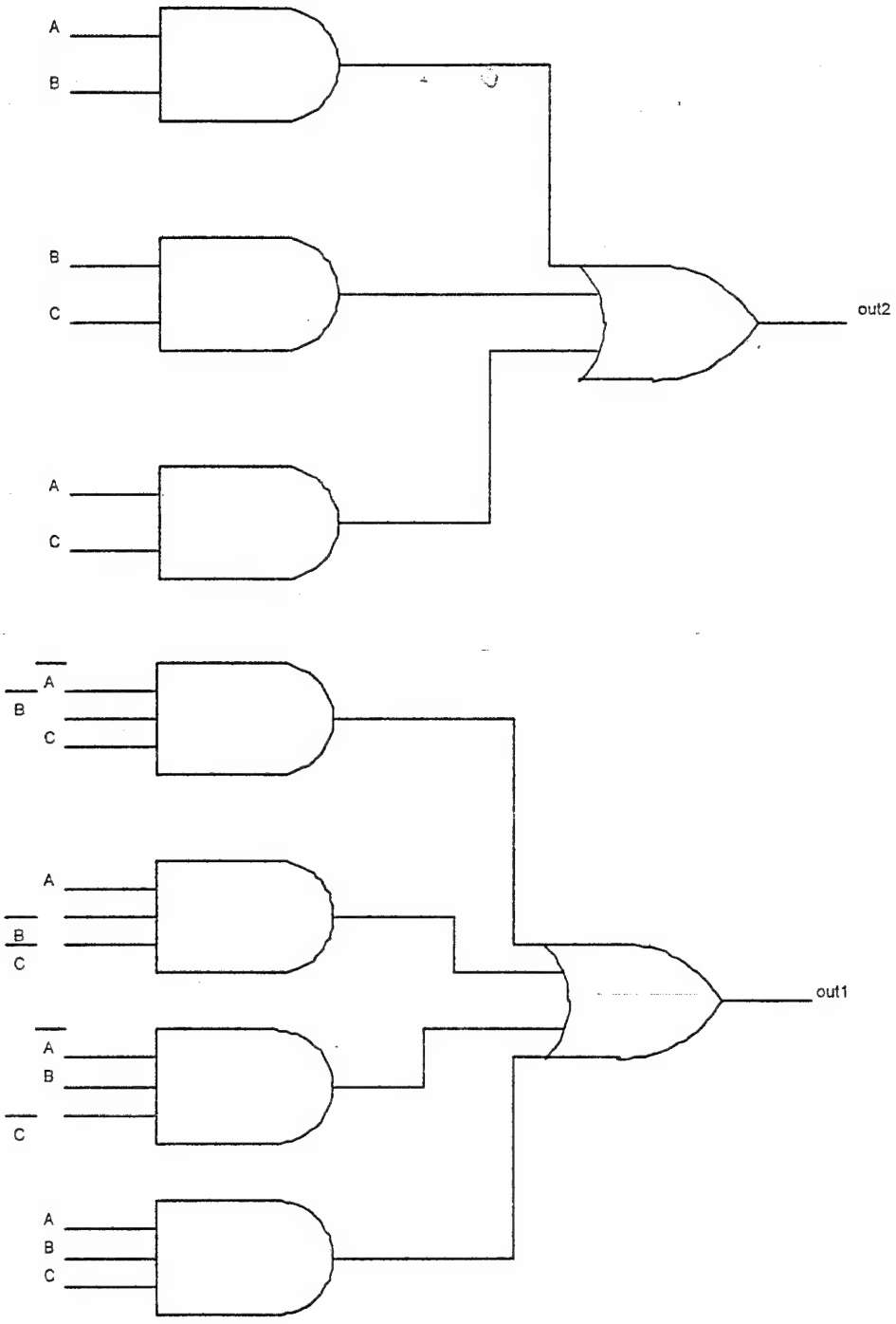


Figure 3(b) 3 input 1's counter

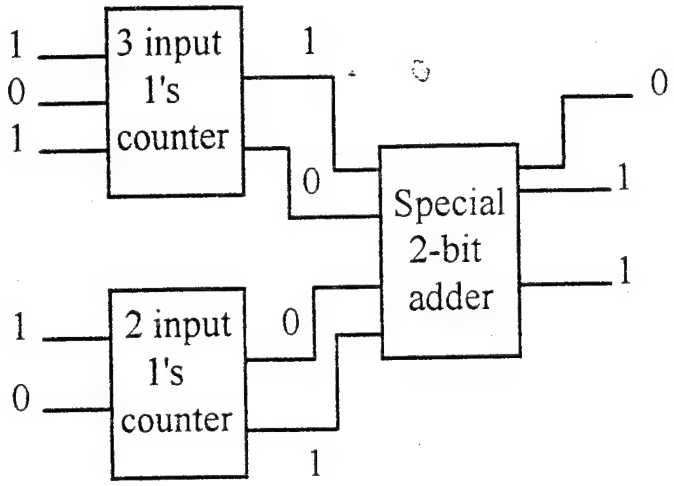


Figure 4

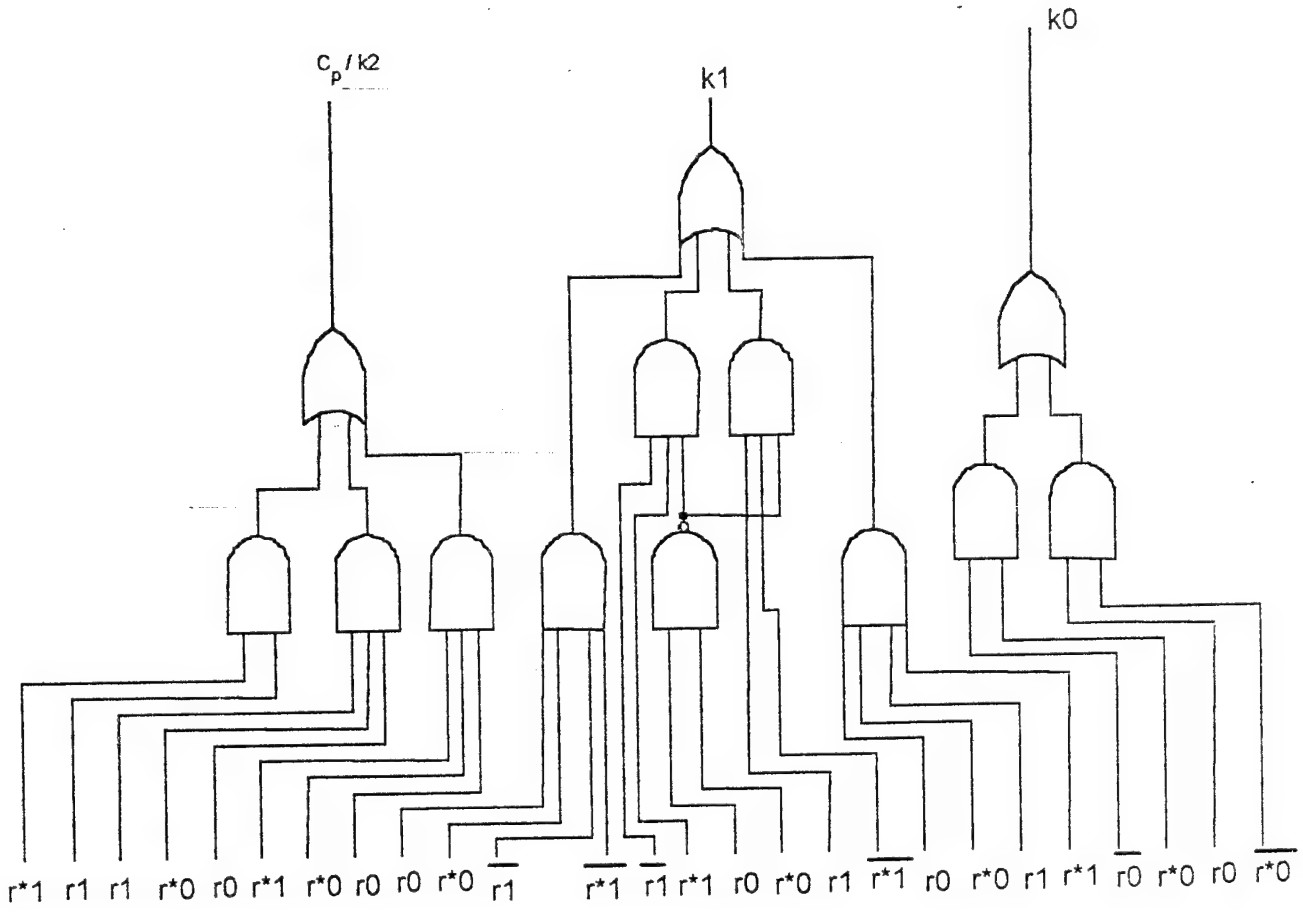


Figure 5(a)

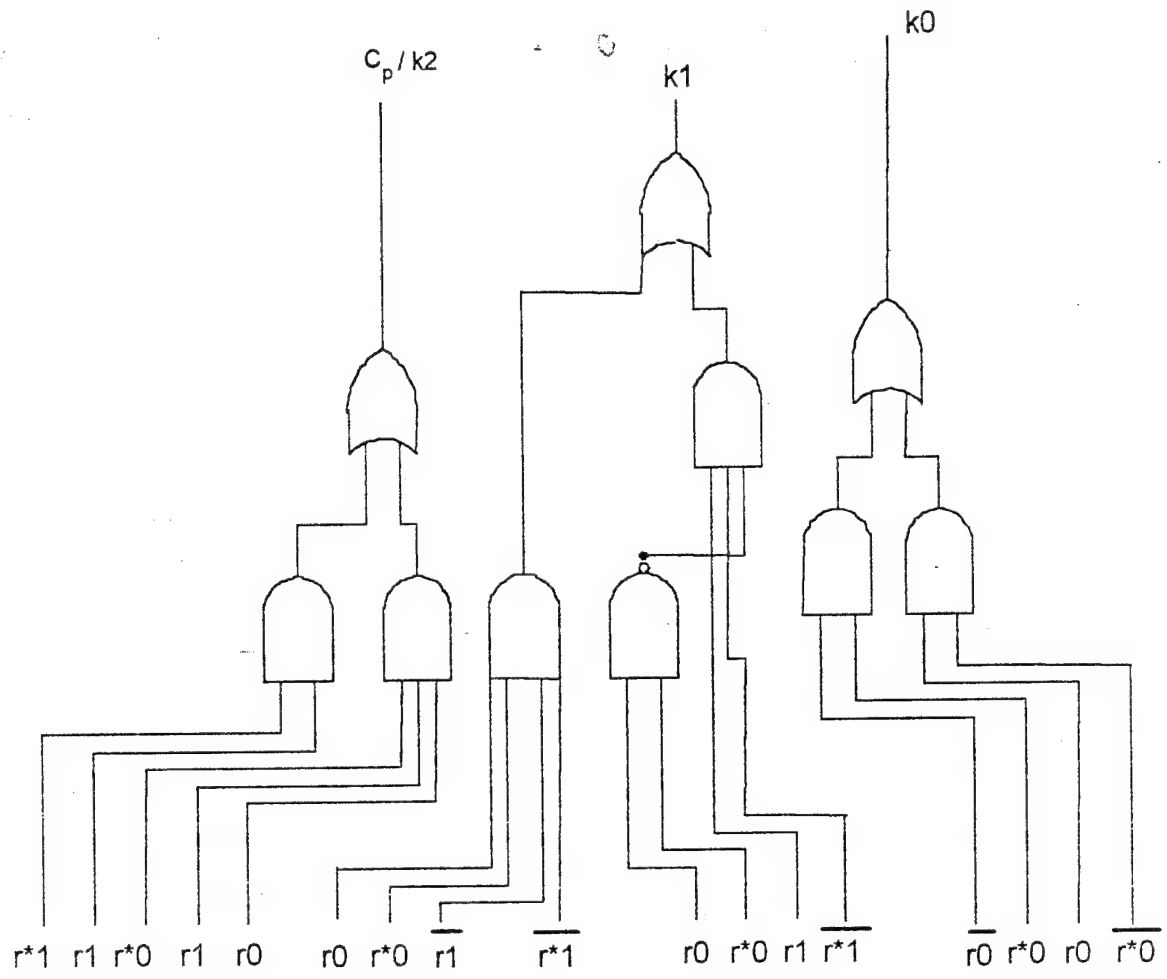


Figure 5(b)

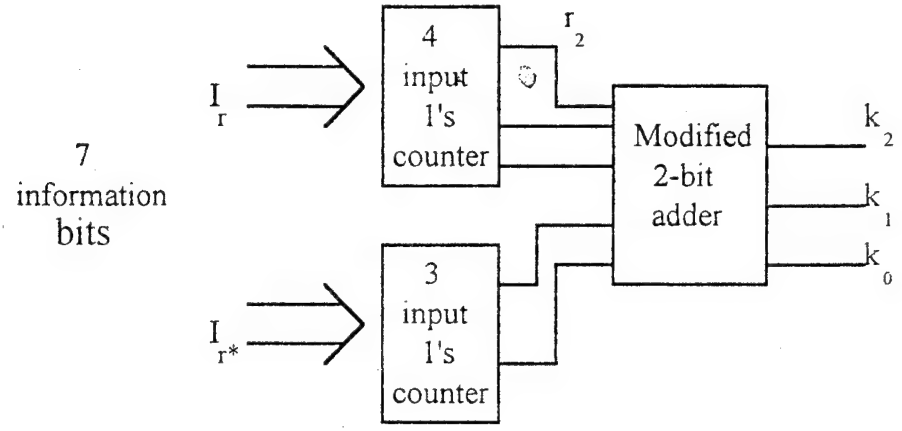


Figure 6

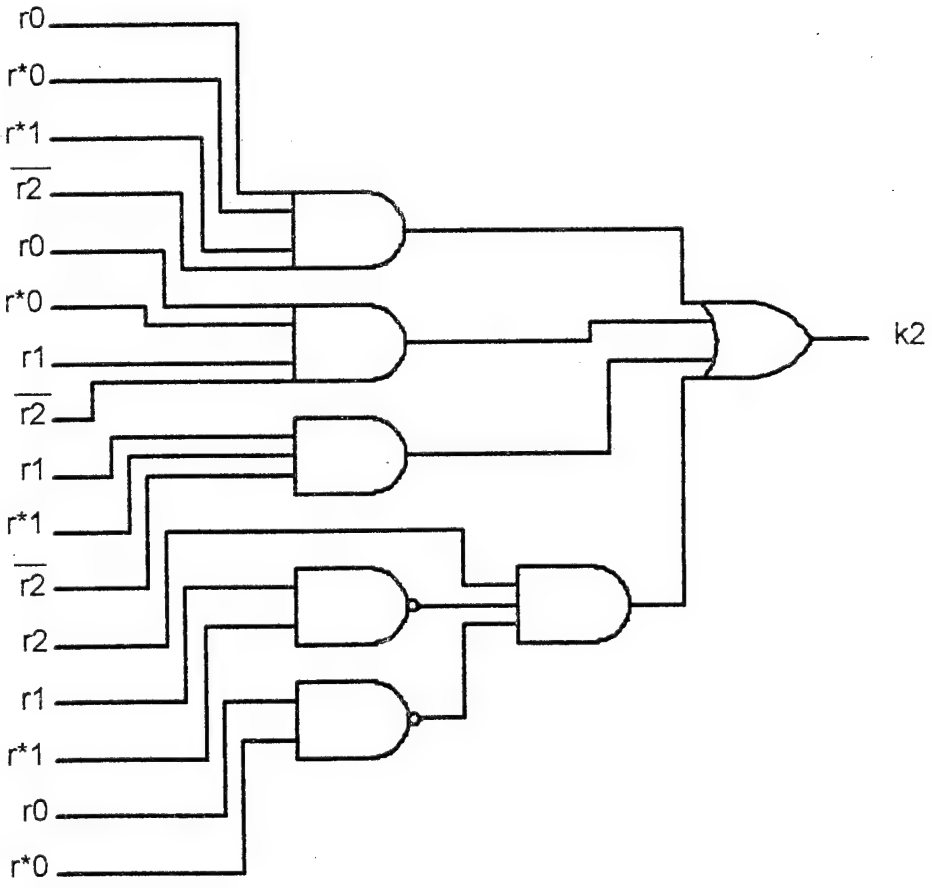


Figure 7

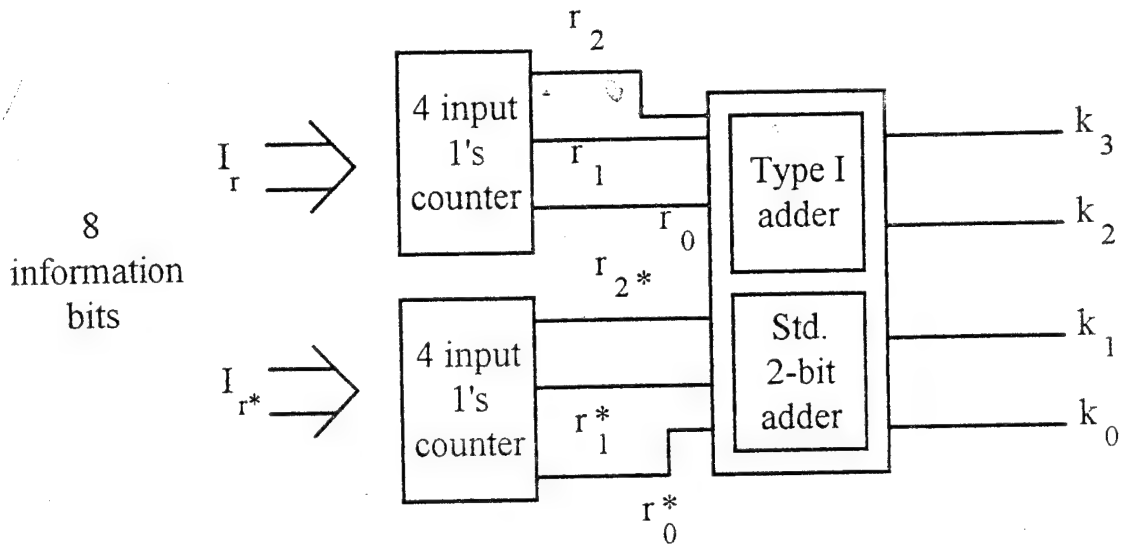


Figure 8

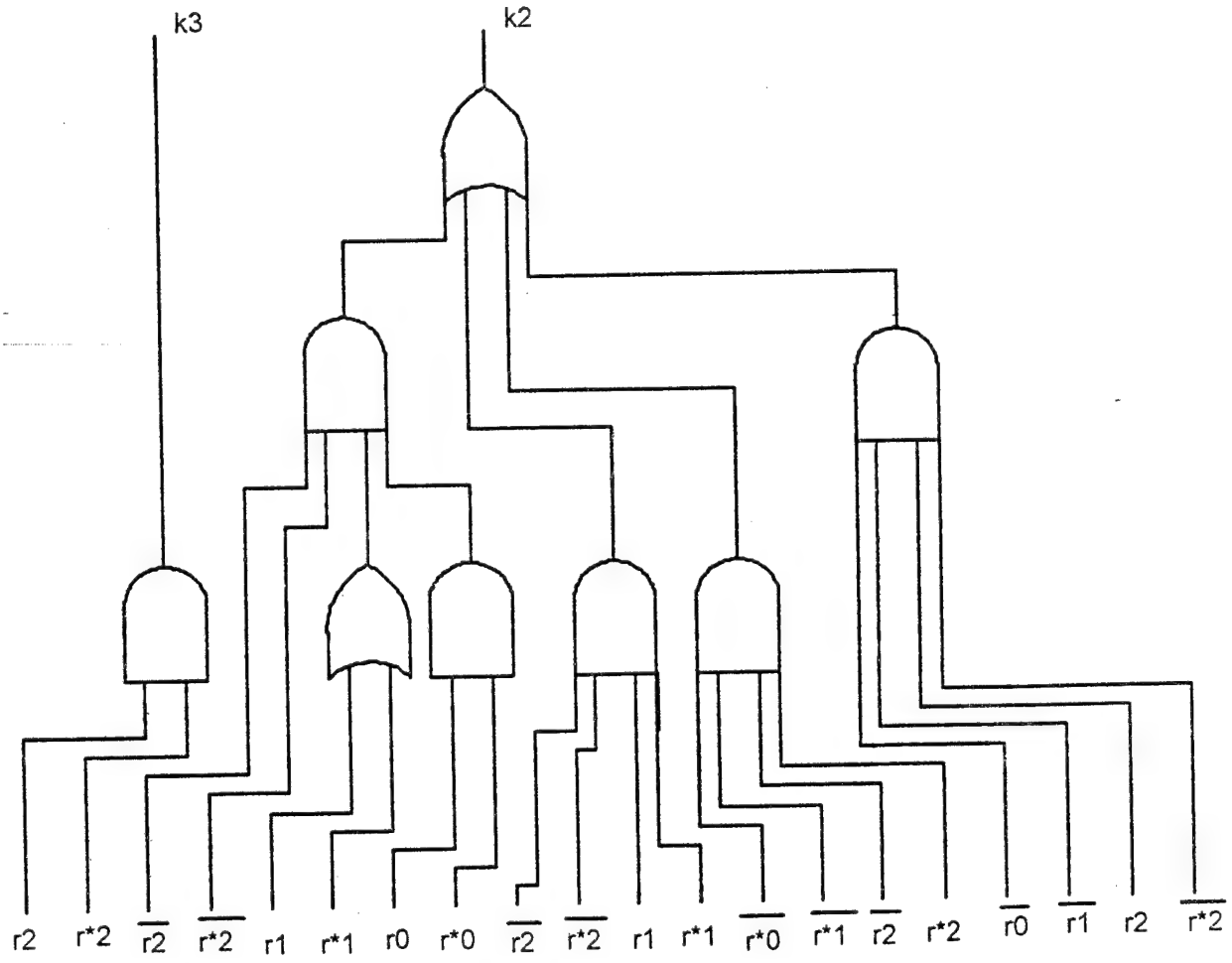


Figure 9

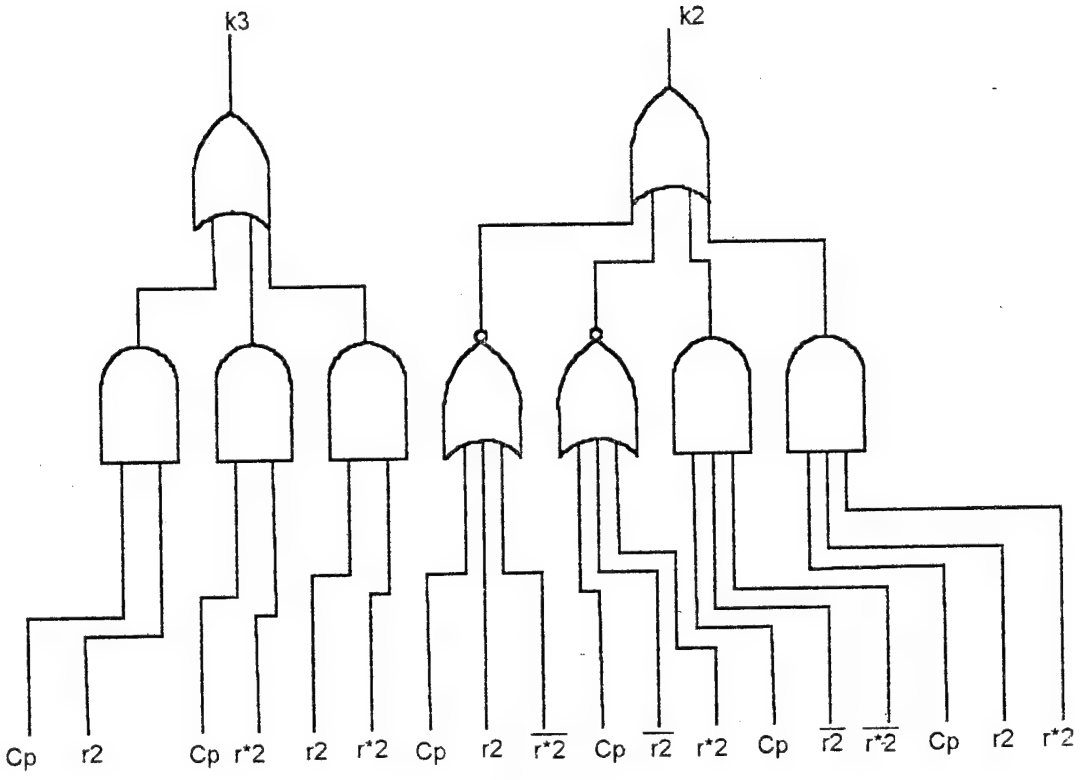

k3
k2
Cp
r2
Cp
r*2
r2
r*2
Cp
r2
$\overline{r*2}$
Cp
$\overline{r2}$
r*2
Cp
$\overline{r2}$
$\overline{r*2}$
Cp
r2
r*2

Figure 10

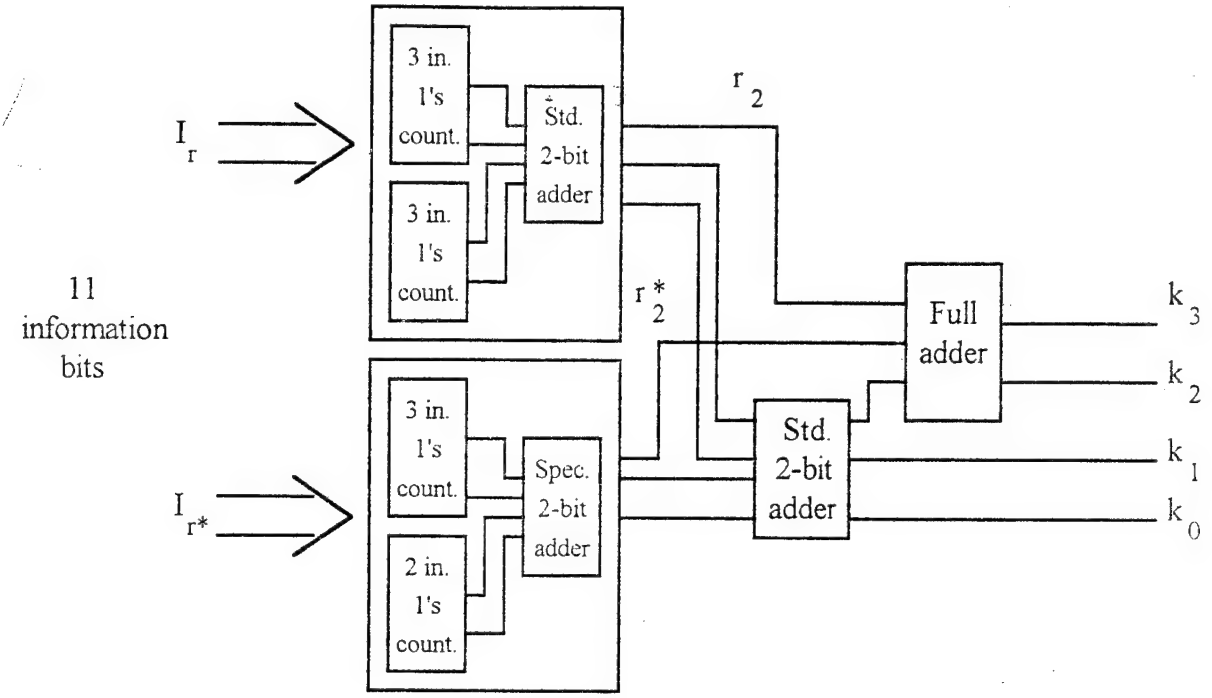


Figure 11

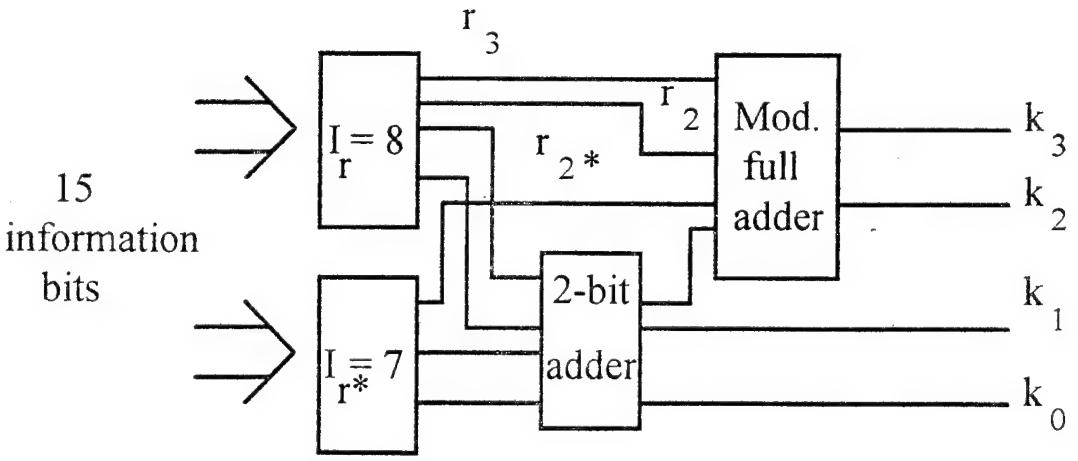


Figure 12

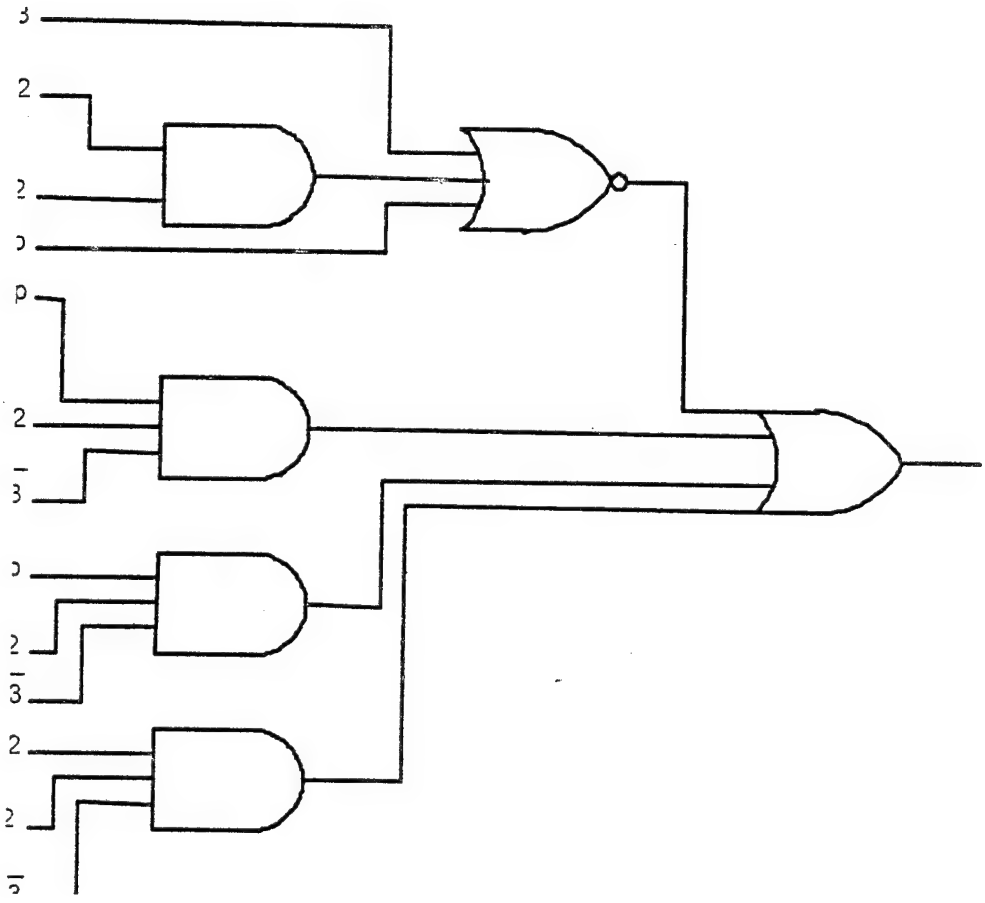


Figure 13

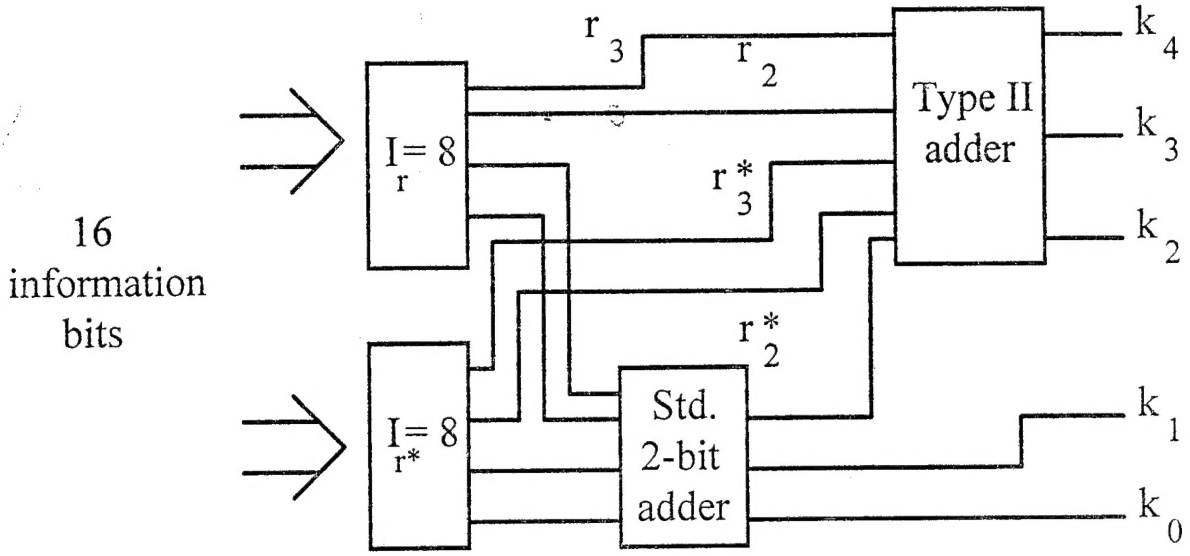


Figure 14

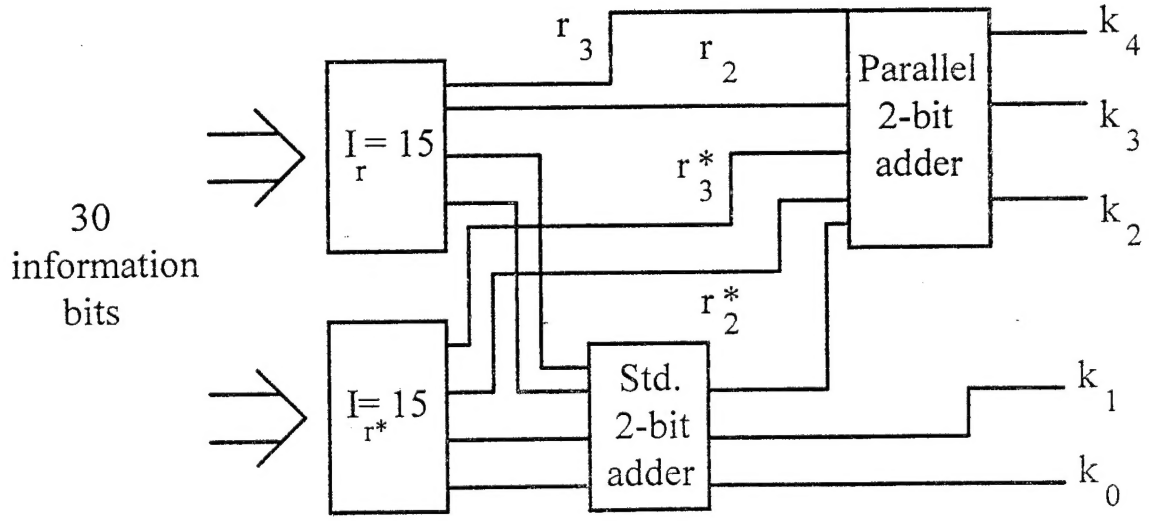


Figure 15

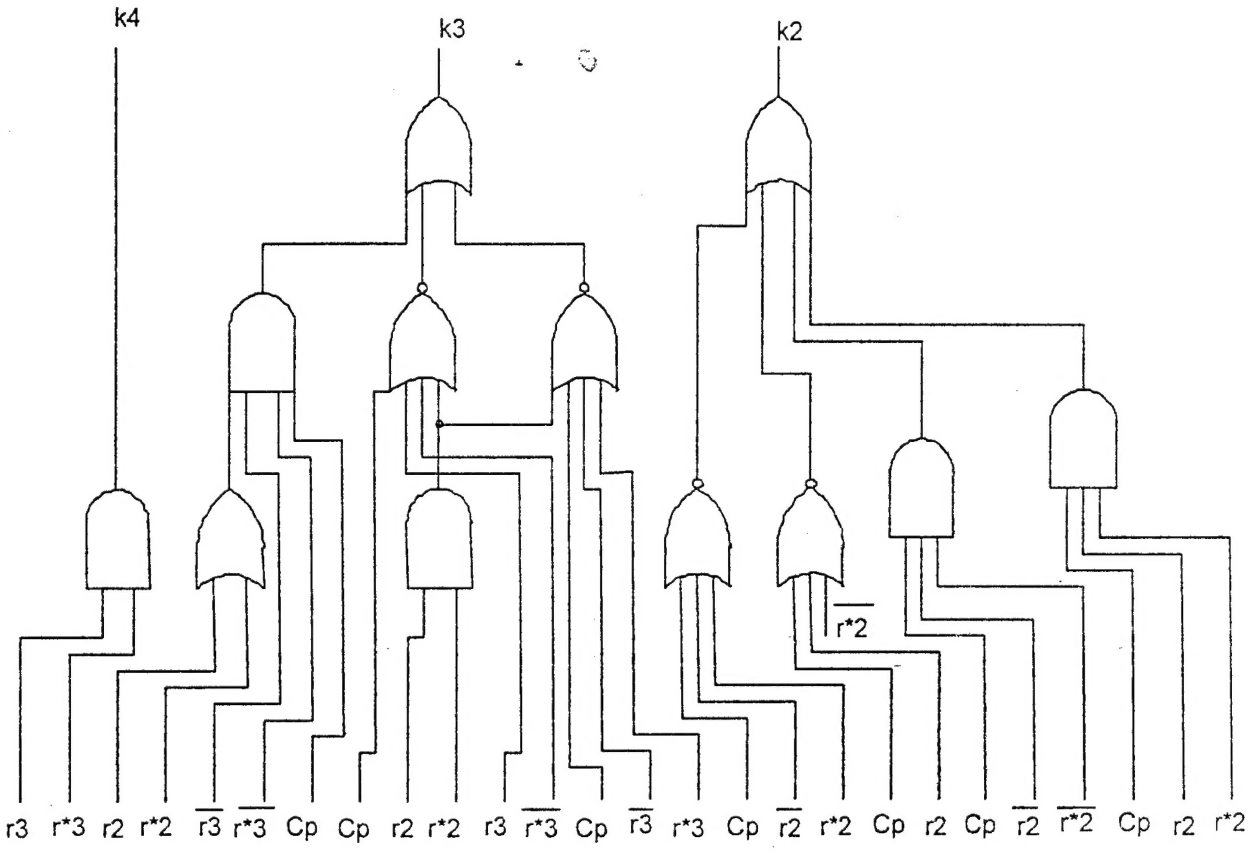


Figure 16

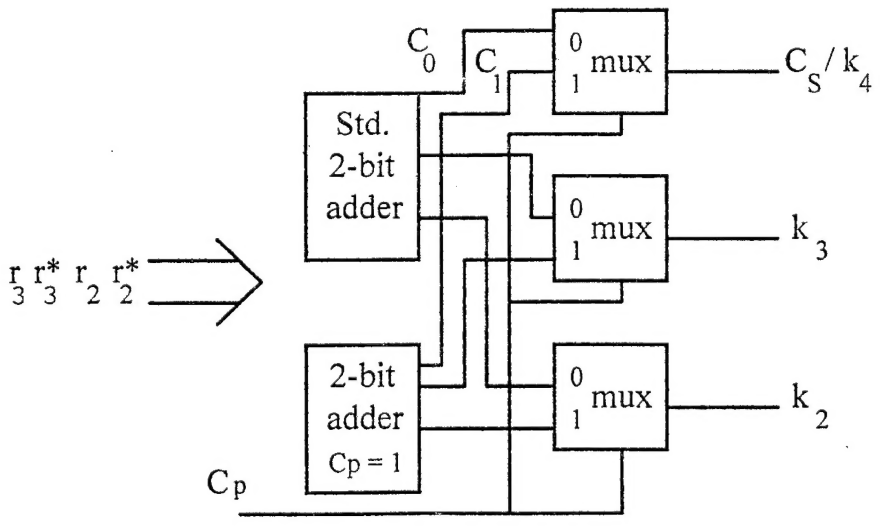


Figure 17

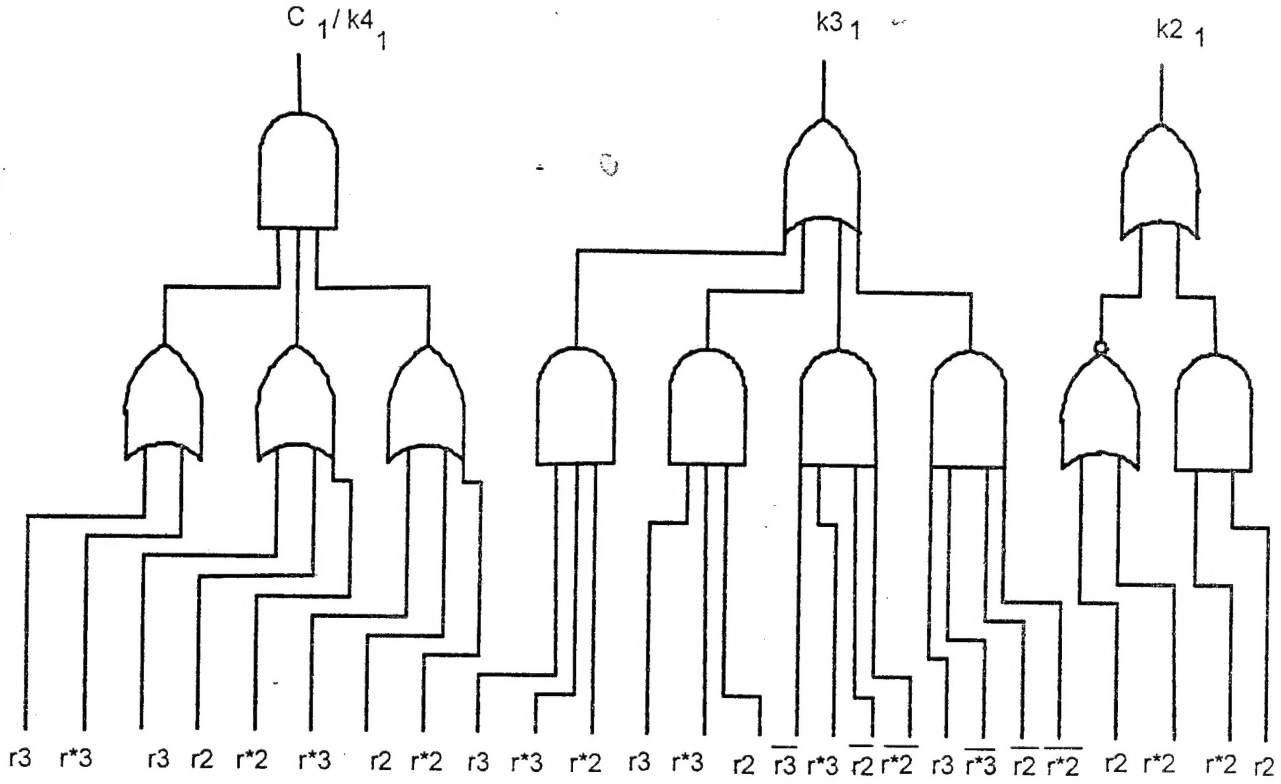


Figure 18

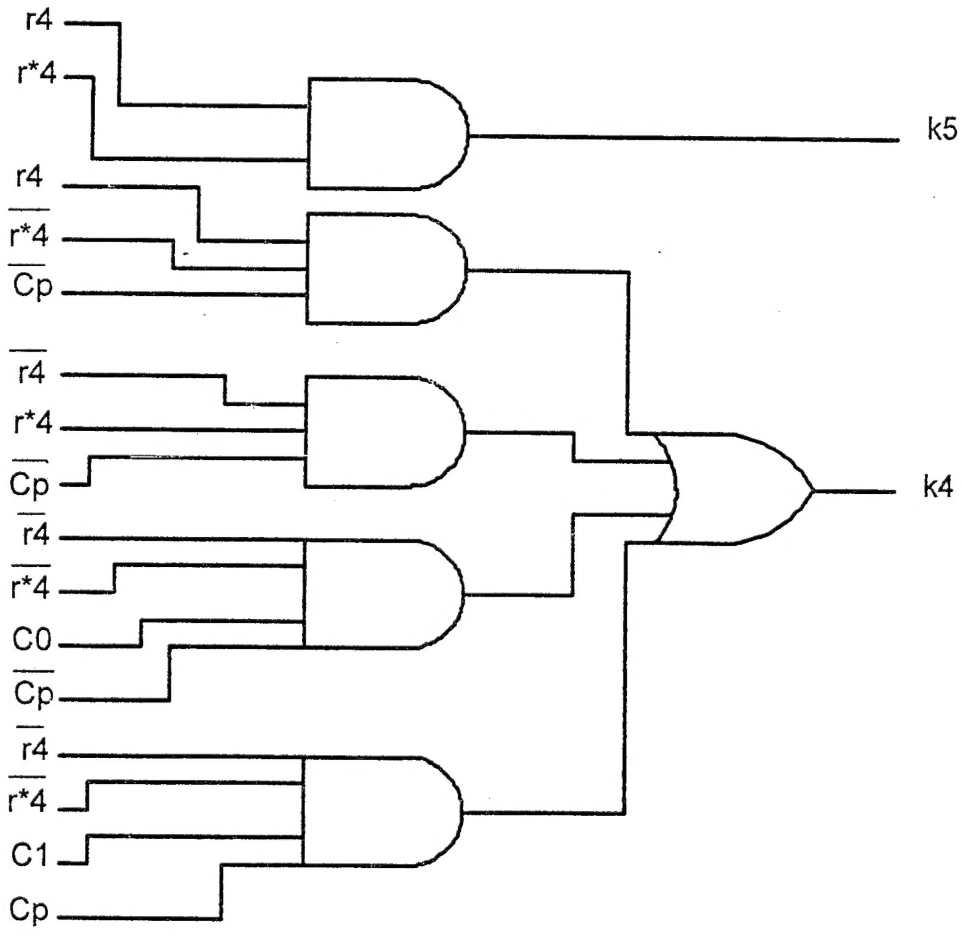


Figure 19

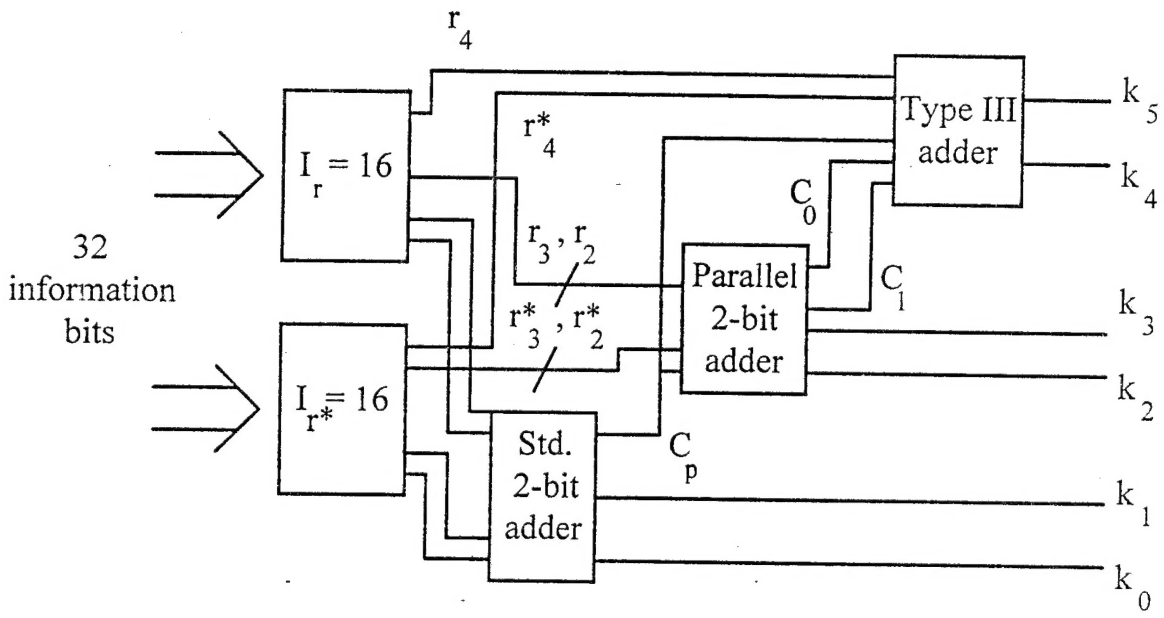

r4
32
information
bits
Ir = 16
Ir* = 16
r*4
r3 , r2
r*3 , r*2
Std.
2-bit
adder
Cp
Parallel
2-bit
adder
C0
C1
Type III
adder
k5
k4
k3
k2
k1
k0

Figure 20